Title: Improving Interactivity of a Parallel and Distributed Immersive Walkthrough Application for Very Large Datasets with Artificial Neural Network-based Machine Learning

Names: Xinlian Liu, Ashish Sharma, Paul Miller, Wei Zhao, Aiichiro Nakano, Rajiv K. Kalia, Priya Vashishta

Affiliation: Concurrent Computing Laboratory for Materials Simulations, Department of Computer Science, Louisiana State University

Postal Address: Department of Physics and Astronomy, Louisiana State University, Baton Rouge, LA 70803-4001

EMail: {liuxl, ashish, pmiller, wzhao, nakano, kalia, priyav}@csc.lsu.edu

Telephone: (225) 578-1342
Fax: (225) 578-5855

Presentation: Xinlian Liu

Keywords:
- Parallel and Distributed Computing
- Scientific Visualization
- Interactivity
- CC4 Algorithm
- Instantaneous Learning

Improving Interactivity of a Parallel and Distributed Immersive Walkthrough Application for Very Large Datasets with Artificial Neural Network-based Machine Learning

Xinlian Liu, Ashish Sharma, Paul Miller, Wei Zhao, Aiichiro Nakano, Rajiv K. Kalia, Priya Vashishta
Concurrent Computing Laboratory for Materials Simulations, Department of Computer Science, Louisiana State University
{liuxl, ashish, pmiller, wzhao, nakano, kalia, priyav}@csc.lsu.edu

Abstract

An instantaneously trained artificial neural network schema is used to improve the interactive speed in very large scale scientific visualization. An instant learning algorithm is adopted to reduce the training time for user behavior analysis in billion-particle walkthrough on an SGI Onyx2 graphics server connected to a PC cluster.

1. Introduction

Scientific visualization provides a deeper understanding of information and data generated by large scale simulations [1]. However, visualization of large datasets remains to be a challenge [2,3,4,5].

Sharma *et al.* [6] implemented a parallel and distributed system that utilizes the computing power of a PC cluster to perform partial pre-rendering to reduce the dataset flowing into the graphics engine. In order to overcome the bottleneck of rather expensive graphics hardware, the computation is split into two parts. The compute-intensive preprocessing including viewer-frustum culling and occlusion culling is performed on the PC cluster, while the graphics processing intensive rendering is carried out on the graphic workstation. This system achieved a nearly interactive rendering speed of about 1 frame per second for a billion particle configuration.

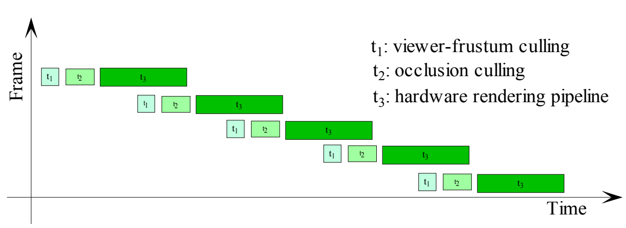

Figure 1: Pipeline overlapping. View frustum culling (t_1) and occlusion culling (t_2) are done on a PC cluster, while hardware rendering (t_3) is done on an SGI graphics server.

To further improve the system performance, it is an intuitive idea to have the two stages pipeline overlapped. For this purpose, there needs to be a mechanism to determine which part of data should be pre-fetched to the second stage in the pipeline, which is the graphics workstation. Figure 1 shows this overlapping scheme.

In order to numerically solve the problem, the target space is divided into a regular 3D grid. User's next move to 26 possible adjacent domains is coded numerically from 1 to 26. Then the problem of predicting next position is converted to that of predicting the value of a time series function.

In this paper, we propose an instantaneously trained neural network schema to improve the interactive speed in extremely large scale scientific visualization.

2. Time Series Prediction

Time series prediction problem is to predict a future value based on knowledge of historical data. Some traditional methods, such as probabilistic method in time series function prediction was discussed in the book by Brown [7]. One of the presumptions of these methods is that the overall distribution of the data conforms to a known probability distribution. However, as the temporal sequence considered in this paper represents the trace of human walking through a 3D space, it is difficult to define such a pattern.

There are alternative methods for time series function prediction, such as Monte Carlo methods [8], and artificial neural networks [9]. Monte Carlo methods usually involve excessive computations, which is not suitable for interactive applications. For neural networks, the obstacle lies in the training time. A prototypical neural network implementation consists of two separate phases: The (usually off-line) training phase and the on-line running phase. The need for off-line training in advance makes neural network unsuitable for tasks that require instant learning from streaming data.

One promising method is the quick training algorithm for radial basis function (RBF) neural networks [10]. It has no local minima, and it can be trained significantly faster than backpropagation networks. However, because of the non-linear shape of the activation function, it is not sufficiently fast for interactive applications.

3. CC4 Algorithm

Kak [11] suggested a unique instant learning neural network based on a corner classification (CC) algorithm. The CC algorithm involves little computation and thus enables instantaneous training. This algorithm is suitable for fast classification problems with binary input data. Several implementations of the CC algorithm have been suggested. One of which, named CC4 combines learning and generalization, and is most suitable for our problem.

A CC4 neural network consists of three layers of binary neurons: The input layer, a hidden layer, and the output layer. The number of input neurons is equal to the length of the input vector plus one bias neuron, which always has the value of 1. The number of hidden neurons is equal to the number of training samples, where each hidden neuron corresponds to one training sample. The activation function of CC4 is: The output neuron outputs 1 when the sum of all weighted inputs is greater than 0 and outputs 0 otherwise. Input layer weights are assigned as:

$$w_j = \begin{cases} 1 & x_i = 1 \\ -1 & x_i = -1 \end{cases}, w_{bias} = r - s + 1$$

For each training vector, if an input neuron receives a 1 $(x_i = 1)$, its weights to all hidden neurons are set to 1; otherwise, they are set to -1. The weights from the bias neuron to the hidden layer neurons are equal to the radius of generalization r minus the summation of '1's in the training vectors plus 1. Output layer weights are assigned as follows: If the training vector produces a 1 at an output neuron, the weight from its hidden neuron to that output neuron is set to 1; otherwise, it is set to -1. Fig. 2 shows the architecture of a general CC4 network.

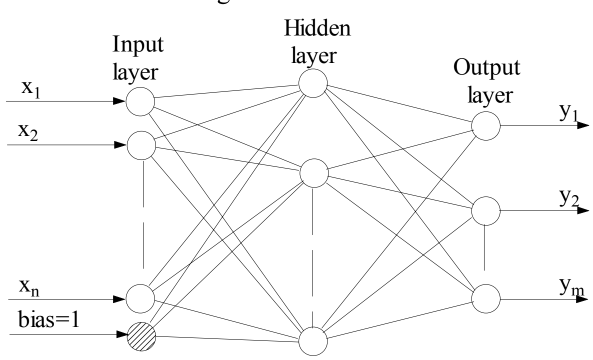

Figure 2: A general CC4 network architecture

A pseudo code for the CC4 training algorithm is given below:

```
// w: input layer weight matrix;
// u: output layer
assign r an appropriate value, i.e. r=2;
for each training vector j do
  begin
    s = 0;
    for each input vector i do
      begin
        if training vector j gets '1'
          then s = s + 1;
        set wi[j];
      end;
    wbias[j] := r - s + 1;
    set u[j];
  end;
```

CC4 algorithm can be used when instant learning is desired, such as on-line intelligent search engine [12], or short term predictions such as daytrade stock values.

4. Billion-Particle Interactive Walkthrough

We have incorporated the CC4 algorithm into the interactive walkthrough system discussed in Sec. 1.

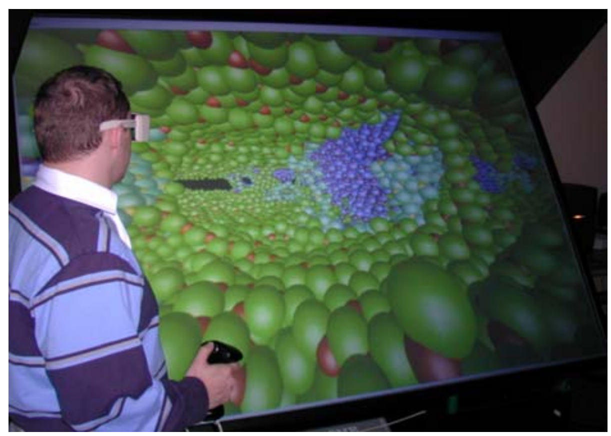

Figure 3: A Scientist is investigating a fracture in a ceramic fiber composite material rendered on an ImmersaDesk virtual environment [13].

In order to improve the average response time, CC4 algorithm is used to predict the next move of the user based on recent previous positions during walkthrough. Using the predicted next move, the program pre-fetches required data while the current scene is being rendered. Therefore, on the next time step, the graphics pipeline can begin rendering immediately without waiting for the data to be fetched through the network. A formal rephrase of the problem is:

$$\begin{cases} \vec{r}_{t+1} = G(\vec{r}_t, \vec{r}_{t-1}, ..., \vec{r}_{t-w+1}, H) \\ H = I(\vec{r}_{t-s+1,} \vec{r}_{t-s+2}, ..., \vec{r}_{t-1}, \vec{r}_t) \end{cases}$$

The predicted position, $\vec{r}_{t+1}$, is determined by the current $(\vec{r}_t)$ and $w-1$ previous positions, $(\vec{r}_{t-1},...,\vec{r}_{t-w+1})$, and knowledge of the walking pattern, *H*, which is obtained by examining the history of s previous positions. The width of testing window, w, and the number of training samples, s, are parameters to be chosen at the training stage to conform to a particular problem.

The entire space is covered with a regular 3D grid, and the smallest unit of this grid forms a cell. The user's position is discretized with the cell. Also, instead of recording absolute positions, relative positions are used, which are represented by directional numerical values as show in Fig. 4. All movements are represented by relative positions with cell 14 being the reference point. For example, number 1 represents a move from position 14 to position 1. A path goes from cell A to cell B (A→C→D→E→F→B) as shown in Fig. 4 will be encoded as a sequence, (15,10,11,23,23), as the path shown as shadow.

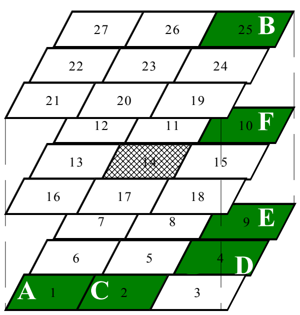

Figure 4: Illustration of the spatial grid. All movements are represented relative to the starting position (cell 14 is the reference cell). A path from A→C→D→E→F→B is represented as a sequence, (15,10,11,23,23).

This predictive pre-fetching acts as 'traffic control', which is similar to the cache management mechanism. It requests future data from the previous culling procedure to feed the graphics rendering pipeline. Before generating a new frame, the rendering pipeline will run a test to check whether the required datasets reside in the memory. If the datasets exist, it will go ahead and perform the rendering. Otherwise, a 'page fault' generates a request for new data.

Because neural network learns by accumulated knowledge over a period of past history, it does not function well until it reaches the point when the historic data is sufficient for extracting such knowledge. We propose a hybrid system, in which heuristic rules are used for prediction in the first few hundred steps and after that, the neural network prediction kicks in. The heuristic rule assumes that the user obeys Newton's laws, which mean it will continue its current moving status, *i.e.,* if moving in one direction, he/she will continue the move in the same direction in the next time step; if the user has been turning, the turning will be continued in the same direction with same speed. A diagram of this hybrid system is shown in Fig. 5.

Given the number of possible movements that can be made during a walkthrough, it is difficult to predict the correct movement every time. In order to speed up overall system performance while tolerate missed predictions, we propose a latency hiding scheme, in which rendering is only performed when the pre-fetching is successful. With this scheme, not all scenes are shown while the user is moving around; however the system performance is improved by taking advantage of the full speed of the graphics rendering hardware.

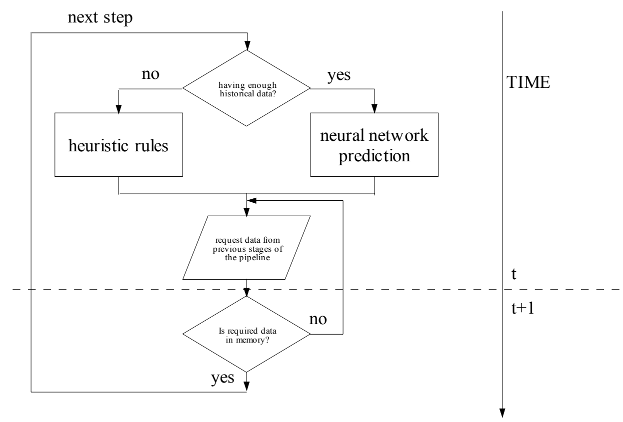

Figure 5: The hybrid predictive pre-fetching scheme.

5. Results

The billion-particle interactive walkthrough project has been implemented on a parallel/distributed platform consisting of an ImmersaDesk virtual environment, an SGI graphics server, and a PC cluster. The ImmersaDesk consists of a pivotal screen, an Electrohome Marquee stereoscopic projector, a head-tracking system, and eyewear kit, IR emitters and a wand with a tracking sensor and a tracking I/O subsystem. A programmable wand with three buttons and a joystick are the primary user interactive devices, which allow interactions between the views and simulated objects. The rendering system is an SGI Onyx2 with two R10000 processors (300 MHz), 4 GB system RAM, and an InfinityReality2 graphics pipeline. The PC cluster used for the pre-processing stage is made of four PCs running Linux 6.2 each with an 800 MHz Pentium III processor and 512 MB RAM.

A scalability test of the system has involved up to a billon atoms. The time to extract and render one scene is nearly a constant function of the number of atoms.

By employing CC4 neural network scheme, the system is more responsive to users' input. Figure 6 shows that the average prediction hit ratio is 34%, which is much higher than that of an alternative heuristic method, 24%. This indicates that in one third of times, the user feels significantly improved system response. Combined with the proposed latency hiding scheme, the system will only render on average about 1 frame per three frames, but the user will experience apparent performance gain in terms of interactivity.

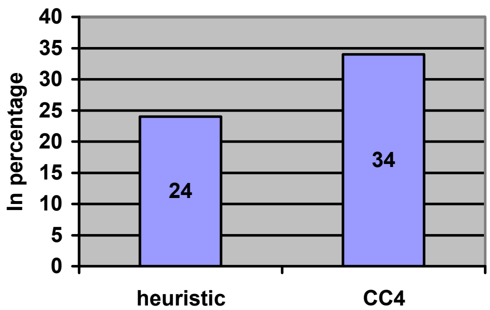

Figure 6: Hit-ratio (in percentage) comparison of CC4 algorithm and heuristic method

6. Summary

We have developed a novel method for interactive control in very large-scale scientific visualization applications. This method uses an instantaneous training neural network CC4 algorithm to track and analyze user behavior and use this information to improve system responding speed. Experimental results show that this method returns better results than guessing or heuristic methods.

7. References

[1] B.H. McCormick, et al., Visualization in Scientific Computing, *Computer Graphics*, 21: 1-14

[2] P.H. Smith and J.V. Rosendale, eds. Data Visualization Corridors: *Report on the 1998 DVC* Workshop *Series*, 1998

[3] C. Bajaj and S. Cutchin, Web based collaborative visualization of distributed and paralle simulation, *IEEE Parallel Visualization and Graphics Symposium*, 1999, San Fransciso

[4] K.L. Ma and D.M. Camp, High Performance Visualization of Time-Varying Volume Data over a Wide-Area Network, *High Performance Networking and Computing Conference*, 2000, Dallas, TX

[5] D. Aliaga, et al., A Framework for the Real-Time Walkthrough of Massive Models, 1998, University of North Carolina: Chapel Hill

[6] A. Sharma, et al., Immersive and Interactive Exploration of Billion-Atom System, *IEEE virtual Reality 2002 Conference*, 2002, Orlando, FL

[7] R.G. Brown, Smoothing, Forecasting and Prediction of Discrete Time Series, *Management and Quantitative Methods Series*, 1963, Englewood Cliffs, NJ: Prentice-Hall

[8] S. Thrun and L. Langford, Monte Carlo Hidden Markov Models, 1998, CMU-CS-98-179, Carnegie Mellon University: Pittsburgh, PA

[9] M.C. Mozer, Neural Net Architectures for Temporal Sequence Processing, *Predicting the Future and Understanding the Past*, A. Weigend and N. Gershenfeld, eds. 1994, Addison-Wesley Publishing: Redwood City, CA

[10] L. Fu, *Neural Networks in Computer Intelligence*, 1994, New York: McGraw-Hill

[11] K.W. Tang and S.C. Kak, A new corner classification approach to neural network training, *Circuits Systems Signal Processing*, 1998, 17(4): p.459-469

[12] B. Shu and S.C. Kak, A neural network based intelligent metasearch engine, *Information Sciences*, 1999, 120: p. 1-11

[13] A. Nakano, et al., Multiscale Simulation of Nanosystems, *IEEE/AIP Computing in Science and Engineering* 3(4), 2001, pp. 56-66